Reusing Proofs when Program Verification Systems are Modified

Bernhard Beckert, Thorsten Bormer,
and Vladimir Klebanov
vladimir@uni-koblenz.de

November 8, 2005

Context: The KeY Project

University of
Karlsruhe (TH)

CHALMERS

Chalmers University
of Technology

University of
Koblenz

www.key-project.org

The KeY System

Components:

- Case tool (Borland Together Architect, Eclipse)
- Spec. authoring tools
- Verification middleware
- Interactive/automated theorem prover

Input
Java program
OCL/JML

Output
Proof in Dynamic Logic

The Problem

Stored proof objects

+

Modified proof system

Claim 1: Affects all proof systems

Changes in...

- Logic Syntax

- The Taclet Language
- Parser/Disambiguation

Changes in...

- Logic Syntax
  - `exists x:int.prop(x)`
- The Taclet Language
- Parser/Disambiguation

Changes in...

- Logic Syntax
  - `exists x:int.prop(x)`
    `\exists java.lang.Object o; prop(o).`
- The Taclet Language
- Parser/Disambiguation

Changes in...

- Logic Syntax
  - `exists x:int.prop(x)`
    `\exists java.lang.Object o; prop(o).`
  - `<program>prop`
- The Taclet Language
- Parser/Disambiguation

Changes in...

- Logic Syntax
  - `exists x:int.prop(x)`
    `\exists java.lang.Object o; prop(o).`
  - `<program>prop`
    `\<prop\>formula`
- The Taclet Language
- Parser/Disambiguation

Changes in...

- Logic Syntax
  - `exists x:int.prop(x)`
    `\exists java.lang.Object o; prop(o).`
  - `<program>prop`
    `\<prop\>formula`
    in order to allow `a<b` in place of `lt(a,b)`
- The Taclet Language
- Parser/Disambiguation

Changes in the Logical Structure of the Rules

$$\frac{\Gamma \vdash ((a > b) \rightarrow \langle \pi \ l = \texttt{true}; \ \omega \rangle \phi) \wedge (\neg(a > b) \rightarrow \langle \pi \ l = \texttt{false}; \ \omega \rangle \phi)}{\Gamma \vdash \langle \pi \ l = a > b \ \omega \rangle \phi}$$

$$\frac{\Gamma \vdash \mathit{if}\ (a > b) \ \langle \pi \ l = \texttt{true}; \ \omega \rangle \phi \ \mathit{else}\ \langle \pi \ l = \texttt{false}; \ \omega \rangle \phi \ \mathit{fi}}{\Gamma \vdash \langle \pi \ l = a > b \ \omega \rangle \phi}$$

Changes in the Logical Structure of the Rules

11:boolean var;	11:boolean var;
12:var=j>0;	12:var=j>0;
13:Update Simplification	13:Update Simplification
14:greater _ than	14:greater _ than
15:concrete _ impl _ 1	15:ifthenelse _ true
16:greater _ than	
17:concrete _ not _ 1	
18:concrete_impl_2	
19:concrete _ and _ 3	
20:var=true;	16:var=true;
21:var=true;	17:var=true;
22:Update Simplification	18:Update Simplification
23:j=0;	19:j=0;
24:Update Simplification	20:Update Simplification
25: { }	21: { }
26:Update Simplification	22:Update Simplification
27:close _ by _ true	23:close _ by _ true
28:Closed goal	24:Closed goal

Change of Prover Interna

- Non-determ. formula/branch ordering
- Non-determ. source model link
- Internal data structure change

Changes in Java Formalization

$$\frac{\Gamma,\ a = null\ \vdash\ \langle \pi\ \texttt{NPE};\ \omega\rangle\phi \qquad \Gamma,\ a \neq null \wedge (i < 0 \vee i \geq a.length)\ \vdash\ \langle \pi\ \texttt{AOBE};\ \omega\rangle\phi \qquad \Gamma,\ a \neq null \wedge i \geq 0 \wedge i < a.length\ \vdash\ \{a[i] := val\}\langle \pi\ \omega\rangle\phi}{\Gamma\ \vdash\ \langle \pi\ a[i]\texttt{=}val\ \omega\rangle\phi}$$

$$\frac{\Gamma,\ a = null\ \vdash\ \langle \pi\ \texttt{NPE};\ \omega\rangle\phi \qquad \Gamma,\ a \neq null \wedge (i < 0 \vee i \geq a.length)\ \vdash\ \langle \pi\ \texttt{AOBE};\ \omega\rangle\phi \qquad \boxed{\Gamma,\ a \neq null \wedge i \geq 0 \wedge i < a.length \wedge \neg storable(val, a)\ \vdash\ \langle \pi\ \texttt{ASE};\ \omega\rangle\phi} \qquad \Gamma,\ a \neq null \wedge i \geq 0 \wedge i < a.length \boxed{\wedge storable(val, a)}\ \vdash\ \{a[i] := val\}\langle \pi\ \omega\rangle\phi}{\Gamma\ \vdash\ \langle \pi\ a[i]\texttt{=}val\ \omega\rangle\phi}$$

Changes in Java Formalization...

Claim 2: ...are inevitable

Remedies:

- Formal rigor? (not rigorous)
- Paying attention?
  - What does `y=x++;` do?
  - What does `x=x++;` do?
- Cross-checking

The Problem (Recap)

Available

Proof system S_1
Proof system S_2
Proof P_1 for S_1

Needed

Proof P_2 for S_2

Claim 3: This is not a problem of (meta-)logics

The Solution

P_1 is correct for S_1
P_2 will be correct for S_2 (guaranteed by S_2)

We are building a proof search procedure.

Our Solution Foundation

Proof Reuse for Deductive Program Verification
[Beckert, Klebanov @SEFM 2004], implemented in KeY

Observation: Every rule application has a focus:

$$\frac{\Gamma,\ b = \mathsf{TRUE} \vdash \langle \pi\ p\ \omega \rangle \phi \quad \Gamma,\ b = \mathsf{FALSE} \vdash \langle \pi\ q\ \omega \rangle \phi}{\Gamma \vdash \langle \pi\ \texttt{if}(b)\ p\ \texttt{else}\ q\ \omega \rangle \phi}$$

1. Identify reusable subproofs
2. Similarity-guided proof replay

Example: Integer Arithmetics in Java

Valid for Java integers

$$\texttt{MAX\_INT} + 1 = \texttt{MIN\_INT}$$
$$\texttt{MIN\_INT}*(-1) = \texttt{MIN\_INT}$$
$$\exists x, y.\ (x \neq 0 \wedge y \neq 0 \wedge x * y = 0)$$

Not valid for Java integers

$$\forall x. \exists y.\ y > x$$

Not a sound rewrite rules for Java integers

$$x + 1 > y + 1 \quad \rightsquigarrow \quad x > y$$

Possible Arithmetics Treatment

- The mathematical way: unsound
- The Java way: very difficult to reason about
- The KeY way:
  1. Show the program correct with math. semantics
  2. Show that no overflow occurs at every step

Example: A Charge Card

```
public static void charge(int credit) {
    try {
        if (balance+credit > maxBalance)
           throw new IllegalArgumentException();
        else
           balance = balance + credit;
    } catch(IllegalArgumentException ex) {
    }
}
```

Invariant property: Is `balance` $\leq$ `maxBalance` always true?

Demo Charge Card

❶ Correct w.r.t. math. semantics? ✔

❷ Correct w.r.t. overflow checking semantics? ✘

❸ Fix bug, now correct ✔

Unaffected proof parts are reused from step to step.

Thank You!

Questions?

TOC

Context: The KeY Project ❖
The KeY System ❖
The Problem ❖
Changes in... ❖
Changes in the Logical Structure of the Rules ❖
Changes in the Logical Structure of the Rules ❖
Change of Prover Interna ❖
Changes in Java Formalization ❖
Changes in Java Formalization... ❖
The Problem (Recap) ❖
The Solution ❖
Our Solution Foundation ❖
Example: Integer Arithmetics in Java ❖
Possible Arithmetics Treatment ❖
Example: A Charge Card ❖
Demo Charge Card ❖
Thank You! ❖